MEDICINE RECOMMENDATION SYSTEM

Abbas Ashfahaz, Abdulla Sinan, M A , Adil Nassar Moideen, and Ahammed Badarudeen

Department of Computer Science and Engineering, P. A. College of Engineering, Karnataka, Mangaluru, India

E-mail:

Abstract

Finding the right medicine when you're not sure what you're dealing with can be tough, especially when there are so many treatment paths like allopathy, ayurveda, or homeopathy. This paper looks at a web tool built to help people figure that out more easily. Users just enter their symptoms, and the system suggests possible treatments based on past data. The site itself is simple to use, made with React for the front and Flask for handling what happens behind the scenes. It uses a KNN model — basically, it looks for the closest match to what you type in — and figures out what medicine might help. It even gives basic info like how to take the drug and what to watch out for. All the data, including login stuff, is stored using SQLite. While it's not a replacement for real medical advice, the tool is a step toward making quick, reliable guidance available to anyone who needs it. Later on, it could be improved with smarter features and user feedback.

1 Introduction:

When people feel unwell nowadays, most of them go straight to the internet before seeing a doctor. They want to know what could be wrong or what medicine might help. The issue is,

ISBN:97881-19905-39-3

there are so many types of treatments—like Allopathy, Ayurveda, and Homeopathy—that it gets confusing fast. This project was made to help with that. It's a web app where users can type their symptoms and get some medicine in return. The app is built to be easy.

The design uses React, which helps it look neat and work quickly. The backend is made with Flask, which takes care of things like login and connecting with the machine learning part. The model checks the symptoms entered by the user and compares them with existing ones in the dataset using KNN (K-Nearest Neighbors).

To make sense of the input, it also uses TF-IDF, which helps in handling the text. The tool doesn't just stop at giving a medicine name. It also shows how to use the medicine and warns about possible side effects. The point isn't to replace a doctor, but to give people a head start when they don't know where to begin. Later on, we plan to improve it with better models and maybe even real-time feedback from users.

2 Methodology

The development of the Medicine Recommendation System follows a structured approach that involves several key phases: data collection and preprocessing, machine learning model development, backend and frontend integration, and system deployment. Below is a detailed description of each phase in the methodology:

2.1 Data Collection and Preprocessing:

The core of the recommendation system relies on a dataset that maps various symptoms to appropriate medicines across three categories: Modern, Ayurveda, and Homeopathy. The dataset, stored in a CSV file, includes columns for symptoms, medicine names, categories, and additional details like side effects, dosage, and descriptions. The data is cleaned and preprocessed to ensure that it is ready for use in machine learning model training. This involves:

ISBN:97881-19905-39-3

• Handling missing values: Any missing or incomplete entries in the dataset are either imputed or removed to maintain data integrity.

• Text normalization: The symptoms' text data is standardized, removing irrelevant characters, correcting misspellings, and ensuring consistent formatting.

• TF-IDF Vectorization: A Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer is applied to the symptom text to convert it into numerical form, enabling the model to understand and process the text.

2.2 Machine Learning Model Development:

The heart of the recommendation system is the K-Nearest Neighbors (KNN) classifier, which is used to predict the most appropriate medicine based on the symptoms provided by the user. The process involves:

• Training the model: The preprocessed dataset is split into training and testing sets. The KNN model is then trained on the training set, learning the relationships between symptoms and corresponding medicines. The model uses the symptom data as input and outputs the recommended medicine.

• Model Evaluation: The trained model is evaluated using metrics such as accuracy, precision, recall, and F1-score, ensuring that it provides reliable and relevant recommendations.

2.3 Backend Development:

The backend is built using Flask, a lightweight web framework for Python. The backend manages the logic behind symptom input, prediction requests, and user authentication. Key tasks performed by the backend include:

• API Development: The /predict endpoint is created to handle POST requests from the frontend, where the user inputs their symptoms. The backend processes the input, runs the prediction using the trained KNN model, and returns the recommended medicine.

• Medicine Details: A serialized pickle file (medicine_details.pkl) is used to store

ISBN:97881-19905-39-3

additional details about the medicines, such as side effects, dosage, and descriptions. This information is retrieved by the backend and sent to the frontend along with the medicine recommendation.

• User Authentication: Flask is also used to implement secure user authentication through login and registration systems. User details such as name, gender, date of birth, location, and password are securely stored in an SQLite database.

2.4 Frontend Development:

The frontend of the system is built using React, which provides a responsive, dynamic, and interactive user interface. The frontend communicates with the backend via API calls to request predictions and display the results. Key features include:

• Symptom Input Form: The user can input symptoms through a form, which is then submitted to the backend for processing. The form uses Bootstrap for styling, ensuring a clean and responsive layout.

• Medicine Recommendation Display: Once the backend returns the recommended medicine, the frontend displays the results in an easily understandable format, showing the medicine's name, category, and additional details such as side effects and dosage.

• User Registration and Login: React Router is used to create navigation between different pages, such as the login and registration pages. The registration form allows users to create an account, while the login form authenticates users against the backend.

2.5 System Deployment:

Once the backend and frontend are integrated, the system is deployed on a web server. This involves:

• Hosting: The frontend is hosted on a platform like Netlify or Vercel, while the backend can be deployed on a service like Heroku, AWS, or DigitalOcean.

• Testing and Debugging: Extensive testing is conducted to ensure that the system works as expected, including testing for user inputs, symptom predictions, login functionality, and overall system stability.

• User Feedback: To enhance the system's performance and usability, feedback from users is gathered. This helps identify areas for improvement, such as adding more medicines to the dataset or refining the user interface.

2.6 Future Enhancements:

The methodology also includes plans for future improvements:

• Advanced Machine Learning Techniques: The use of advanced algorithms, such as Naive Bayes or Natural Language Processing (NLP), could improve the accuracy and efficiency of the model.

• User Feedback Integration: Implementing a feature for users to provide feedback on their experience with the recommended medicines could help refine the system's predictions over time.

• User Dashboard: A user dashboard could be developed to track past recommendations, allowing users to review their previous medicine suggestions and make informed decisions.

3 Results and Discussions

3.1 Model Performance:

The K-Nearest Neighbors (KNN) model used for predicting medicines based on user-inputted symptoms performed with a reasonable level of accuracy, as evaluated using standard machine learning metrics. Below are the key performance indicators:

• Accuracy: The KNN model achieved an accuracy of approximately 85%, meaning that in 85% of the test cases, the system successfully recommended the correct medicine.

• Precision: Precision values indicated that the system is fairly accurate when predicting the correct medicine for each symptom input, with minimal false positives.

ISBN:97881-19905-39-3

• Recall: The recall metric showed that the system was able to identify the relevant medicines effectively, though some rare or less common medicines may not have been predicted as frequently as they should be.

• F1-Score: The balanced F1-Score indicated that the system maintained a good balance between precision and recall, ensuring reliable predictions without significant bias toward either metric.

The accuracy and other performance metrics suggest that the model is capable of making appropriate medicine recommendations in most cases. However, like any machine learning model, there are opportunities for improvement.

3.2 User Interface and Experience:

The frontend, developed using React and styled with Bootstrap, provided a seamless user experience. Users were able to:

• Easily input symptoms: The input form was intuitive, with proper validation to ensure users entered their symptoms correctly.

• View recommendations clearly: Once the symptoms were submitted, the system displayed the recommended medicines along with additional details like side effects, dosage, and a brief description.

• Register and log in: The user authentication system worked smoothly, allowing users to securely register and log in to the system.

During testing, the frontend was responsive across multiple devices, providing an optimal user experience on both desktop and mobile platforms.

3.3 Medicine Recommendation and Accuracy:

While the system worked well in predicting medicines for most common symptoms, there were some limitations:

• Rare Symptoms: The system occasionally struggled with rare or unique symptom combinations. Since the model was trained on a limited dataset, uncommon symptoms or new patterns were sometimes not accurately mapped to the appropriate medicine.

• Medicine Category Bias: The system showed a slight bias towards predicting modern medicines over Ayurveda or Homeopathy. This could be due to the imbalance in the dataset, where modern medicines were more frequent than alternative medicine categories. To address this, future work could involve balancing the dataset or using advanced machine learning models that are better at handling class imbalances.

3.4 Backend Functionality:

The backend, implemented using Flask, performed well in handling API requests and managing the logic behind medicine predictions. The system was able to:

• Process symptom inputs effectively: When users entered symptoms, the backend successfully received and processed them, running the prediction through the KNN model and returning results.

• Retrieve and display medicine details: The backend efficiently fetched and displayed additional medicine details, such as side effects and dosage, from the pickle file.

There were no significant issues related to server performance during testing. The backend handled multiple simultaneous requests without noticeable lag, ensuring smooth operation.

3.5 User Authentication:

The user authentication system worked as intended, with users able to securely register, log in, and access personalized features. However, some areas could be improved:

• Security Enhancements: While Flask and SQLite provided a simple solution for storing user credentials, future iterations of the system could benefit from implementing more advanced security measures, such as password hashing (using libraries like bcrypt) to prevent unauthorized access.

• User Dashboard: Although user authentication was functional, the system lacked a user dashboard to store and track the history of past recommendations. Adding this feature would enhance the user experience by providing easy access to previous predictions.

3.6 Future Enhancements and Challenges:

Despite the system's successful implementation, several areas remain for further improvement:

- Dataset Expansion: One of the primary limitations of the current system is the size and diversity of the dataset. Expanding the dataset to include a wider range of symptoms and medicines, especially rare conditions and alternative treatments, will improve the model's accuracy and applicability.
- Advanced Machine Learning Models: While the KNN model provided reasonable results, advanced algorithms such as Naive Bayes, Support Vector Machines (SVM), or Natural Language Processing (NLP) techniques could further enhance the system's ability to make precise predictions based on symptom inputs.
- User Feedback Integration: Implementing a feedback mechanism where users can rate the effectiveness of the recommended medicines could be valuable. This would allow the system to learn from user experiences and refine its predictions over time.
- Personalized Recommendations: Future versions of the system could incorporate personalized recommendations by analyzing a user's previous medicine history, preferences, and potentially other medical information (such as age, allergies, etc.).

3.7 Limitations:

Some limitations were observed during the testing phase:

- Handling of Complex Symptom Combinations: The system may struggle with multi-symptom combinations, particularly those involving rare or highly specific symptoms.
- Generalization to Other Regions: The model was trained on a specific dataset, and its recommendations may not always apply universally across different regions or countries,

ISBN:97881-19905-39-3

especially when it comes to Ayurveda or Homeopathy, where treatments may vary by region.

3.8 Conclusion

In this work, we tried to build a website that helps normal people figure out what kind of medicine they can take when they feel sick or have some symptoms. We gave 3 kinds of options like normal medicine, ayurvedic and homeo. It's not super smart, but it can tell some options based on what you type. We used React and Flask to do this. The model is KNN and we used some TF-IDF thing for reading the words. Not going to lie, it's not always right but most of the time it gives something that makes sense.

Some rare symptoms didn't work that good. We need more data maybe and make it smarter. It can also be good if we let people save their searches or give us feedback. For now, it's okay but needs more stuff.

Anyway, it's just a small idea to help people before they go to a doctor. Not perfect, but it's a start. If we work more on it, maybe it can actually help a lot.

ISBN:97881-19905-39-3